wBasics Of CPP:

1.#include<iostream> //Header to include cin and cout

#include<math.h> //to include math operations

using namespace std;

***void Takeinput(){***

***#ifndef ONLINE\_JUDGE***

***freopen("input.txt", "r", stdin);***

***freopen("output.txt", "r", stdin);***

***#endif***

***} //To take input in VScode***

int main(){

Takeinput() //Function call

int n;

std::cin >> n; //std 🡪 namespace; we can include below the header then no need to specify here.

std::cout<<n;

std::cout<<”Hello World”;

}

2. int main(){

Takeinput()

int a=10; //integer

float f=2; //float

double d=2.2453253;

char c=’a’;

string s=”Indhu”;

bool hello=true; //false

}

3. //Reading inputs for different datatypes

int main(){

int a,b;

float c;

cin >> a>>b>>c; //Input can be in single line or multiple lines

cout<<a<<” “<<b” “<<c;}

int main(){

char c;

cin >> c; //Input can be in single line or multiple lines

cout<<c;

string s; //it will take input upto only space for ex: “Indhu hii”; then input taken will be only “Indhu”

cin >> s;

cout<<s;

string withspace;

getline(cin,withspace);

cout<<withspace;

}

4. //If – Else

int main(){

int n;

cin >> n;

if(n&1){

cout<<”Odd”;}

else{cout<<”Even”;}

}

5. Operators:

Arithmetic: +, -, \*, %, /

While division operation if 2 operands are int then only output will be integer else it will be float.

If 2 operands x,y, Modulus operation will result will range in between 0 to y-1. Clock, Rotation related problems can be deal with modulus operator.

1!=1

2!=2

3!=6

4!=24

5!=120

6!=720

7!=5040

Assignment operators: Used for assigning =,+=,-=,\*=,/=

Relational operators: ==,>,<,>=,<=,!=

Logical operators: &&,||,and,or

Bitwise operators: &, |, ^, >>,<<

Right shift >> in the sense it divide the number by 2

Left shift << in the sense multiple by 2

&operation with 1 then it will give whether the number is odd or not

6. Loops:

for(int i=0;i<=100; i++)

we can run the loop on char also

for(char c=’a’; c<=’z’; c++)

while(condition)

Ex:

char c=’a’;

While(true){

cout<<c;

c++;

if(c<=’z’){

break;}}

Before using the below functions declare **#include <algorithm>** header

\*find method arguments are starting index, ending index, key(we wants to search) it works in order of n

int main(){

Takeinput();

int n=5;

int a[n]={12,33,46,75,22};

int key;

cin>>key;

auto id=find(a,a+n;key); //auto will automatically takes the data type we can also declare it as int \*id

int ind=id-a; //To find index of key specified

if(ind>=n){

cout<<”Not Found”};

else cout<<ind;}

binary\_search method: For sorted array binary\_search can be used

int main(){

Takeinput();

int n=5;

int a[n]={1,2,3,4,5};

int key;

cin>>key;

bool state=binary\_search(a,a+n;key); //this method gives either 0 or 1 as output i.e., Boolean datatype

cout<<state;}

lower\_bound & upper\_bound:

lower\_bound method gives the address of key specified if it exists in array else it gives next highest key.(>=) if exists gives the address of same else gives next greater key address

upper\_bound method gives address of strictly next greater key of specified key.

Syntax: upper\_bound(starting address, ending address,key)

In case of repeated keys it gives first key address.

It works as find method.

sort method:

sort(first address, last address) //sorts the array

#include<bits/stdc++.h> //includes all library files

#include <string>

Strings:

string s;

cin >> s; //It will take only upto space

getline(cin, s); // To take with space

Functions included in strings:

empty(): To check whether the string is empty or not ex: s.empty();

append(): To append any other string to existing string ex: s.append(“Hi”);

clear(): we can clear string ex: s.clear();

String can be compared

string s2=”Apple”;

string s3=”Mango”;

if(s2>s3){cout<<s3;}

else cout<<s2;

compare(): we can also use compare for comparing 2 strings. It gives lexicographically smaller.

Ex: s2.compare(s3);

erase(): to erase only specified portion from string. Syntax:s2.erase(first index, first index+len(word))

ex:

string s1=”I love myself”;

s1.erase(2,2+3); // to delete “love” from string

we can use find function in strings also.

string s2=”Nakka”;

for(auto it:s2){

cout<<it;}//To iterate the elements in string

\*vector\*:

vector<int> vectorname;

push\_back(); //method for pushing elements to vector

* Vector<int>B={10,3,43,54,64};

For taking vector input from user.

* vector<int> g1(n);

for(int i=0; i<n; i++){

cin>> g1[i];}

* vector<int> g1;

for(int i=0; i<n; i++){

int x;

cin>>x;

g1.push\_back(x);}

\*map\*:

Syntax: map<datatype(key), datatype(value)> mapname;

Ex:

map<int,int>mapp;

int D[5]={1,2,3,4,5};

for(int i=0; i<D.size(); i++){

mapp[D[i]]++;}// No need of initializing we can directly increment the value

for(auto it:mapp){

cout<<it.first<<” “<<it.second<<endl; //to access map

methods are present

Division map,…

Sequence Containers:

Vector: Dynamic Array;

push\_back(), emplace\_back(): To push elements

begin(): to get the starting address of vector => vectorname.begin();

end(): to get next address of last element in vector => vectorname.end();

to sort vector: sort(vectorname.begin(), vectorname.end());

for knowing size: vectorname.size();

To iterate loop on vector:

for(auto it = A.begin(); it!=A.end(); it++){

cout<<\*it;}

for(auto i : A){

cout<<i;}

pop\_back(): To delete last element => vectorname.pop\_back();

insert(): To insert a element in between 2 elements => vector.insert(vector.begin()+1, 100);

erase(): To erase elements => vect.erase(vect.begin()+1);//single value or vect.erase(vect.begin(),vect.begin()+4);//Multiple values

It operates on address of elements in vector.

reverse(): to reverse a vector => reverse(vect.begin(), vect.end());

rbegin(): last element address => vect.rbegin();

sort(vect.begin(), vect.end(), greater<int>()); //To sort in descending order

sort(vect.rbegin(), vect.rend()); //To sort in descending order.